Claims

Sub D1

1. (Currently amended) In a multitasking operating system that uses virtual memory to share physical memory among concurrently executing application programs, a method for controlling allocation of physical memory comprising:

in response to a call from an application program, other than an operating system, to group specified code or data in a group, creating a structure to group the code or data specified by the application;

monitoring for a not-present interrupt generated in response to request to access any part of the code or data in the group; and

when the not-present interrupt occurs for a unit of memory in the group, loading all of the code or data in the group that is not already in physical memory into physical memory from secondary storage at one time, including loading the unit of memory for which the not-present interrupt has occurred and all other units of memory used to store the code or data in the group.

2. (Original) The method of claim 1 wherein the structure includes a linked list structure that links together code or data stored at non-contiguous portions of virtual memory.

3. (Original) The method of claim 2 wherein the structure links pages of memory associated with the non-contiguous portions of code or data.

4. (Original) The method of claim 1 further including:
repeating the steps of claim 1 for additional groups of code or data specified by the application.

5. (Original) The method of claim 4 further including:

repeating the steps of claim 1 for a group of code or data for another concurrently executing application such that more than one concurrently executing application program has specified at least one group of code or data to be treated as a single piece of memory for loading into physical memory in response to a not-present interrupt.

6. (Original) The method of claim 1 further including:

when the not-present interrupt occurs, checking whether the interrupt has occurred for a unit of memory in the group by evaluating whether an address of the memory request for which the interrupt occurred is within a series of non-contiguous memory addresses of the group.

7. (Original) The method of claim 1 further including:

tracking memory accesses to units of memory in the group together such that when a unit of memory in the group is accessed, all of the units of memory in the group are marked as accessed; and

determining which portions of physical memory to swap from physical memory to secondary storage by determining which units of code are marked as accessed, such that units are selected to be swapped from physical memory to secondary storage based on frequency of use or how recently the units of code have been accessed.

8. (Original) The method of claim 7 further including:

in response to a call from an application program to group specified code or data in a second group, creating a second structure to group the code or data specified by the application;

tracking memory accesses to units of memory in the first and second group such that when a unit of memory in both the first and second group is accessed, all of the units of memory in the first and second group are marked as accessed and the unit of memory in both the first and second group is marked as being accessed twice.

9. (Original) The method of claim 8

when a block of code or data shared between two or more groups is accessed, marking the block as being accessed n times where n is the number of groups that share the block.

10. (Original) A computer-readable medium storing instructions for performing the steps of claim 1.

11. (Canceled)

12. (Canceled)

13. (Canceled)

14. (Canceled)

15. (Canceled)

16. (Canceled)

17. (Canceled)

18. (Canceled)

19. (Canceled)

20. (Currently Amended) A computer-readable medium having stored thereon a data structure comprising:

a series of data fields forming a group for indicating ~~representing~~ blocks of code or data associated with an application to be treated as a single unit for purposes of virtual memory management, the data fields including a list of memory addresses of the blocks and sizes of each block in the list;

wherein the data structure is evaluated in a data processing operation to load each of the blocks into physical memory whenever a not-present interrupt is generated for any memory address referring to a location included in one of the blocks.

21. (New) The computer readable medium of claim 20, wherein the list of memory addresses is an array of pointers to blocks of memory to be placed in the group.

22. (New) The computer readable medium of claim 20, wherein the sizes of each block in the list is indicated in an array of parameters.

23. (New) The computer readable medium of claim 20, wherein the data structure is used to derive a linked list structure for keeping track of pages used to store the code or data associated with the group as specified by the application.

24. (New) The method of claim 1 further comprising, in response to a second call from the application program to further add units of memory to the group, adding the units of memory to the data structure as specified by the application.

25. (New) The method of claim 1 further comprising, in response to a second call from the application to delete specified units of memory from the group, deleting the units of memory specified by the application from the data structure.

26. (New) The method of claim 1 further comprising, in response to a second call from the application to destroy the group, destroying the data structure previously used for creating the group.

27. (New) In a multitasking operating system that uses virtual memory to share physical memory among concurrently executing application programs, a virtual memory management system comprising:

means for creating a data structure to group code or data specified by one of the concurrently executing applications, in response to a call from the application to create a group of specified code or data;

means for monitoring for a not-present interrupt generated in response to a request to access any part of the code or data in the group; and

means for loading all of the code or data in the group that is not already in physical memory into physical memory from secondary storage at one time, including means for loading the unit of memory for which the not-present interrupt has occurred and all other units of memory used to store the code or data in the group, when the not-present interrupt occurs for a unit of memory in the group.